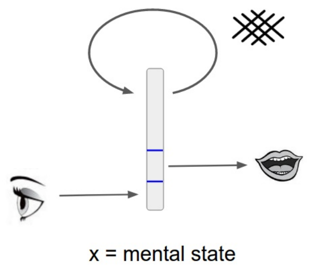
Research progress

甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

April 10, 2017

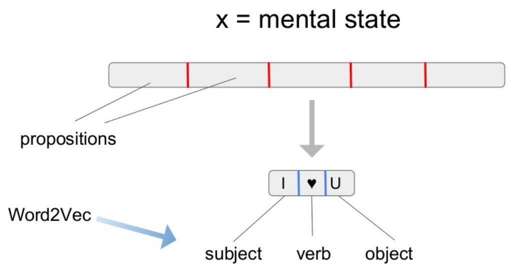
This is the basic architecture:



(1)

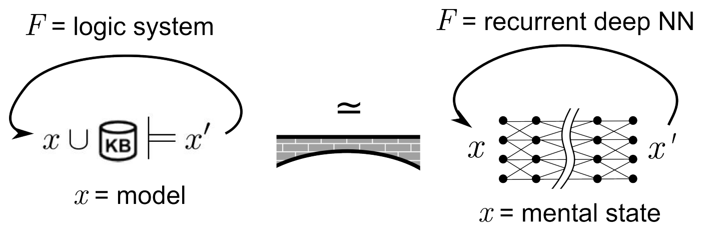
where I use the symbol ⋙ for a neural network.

The mental state is a vector $\boldsymbol{x}$ which should possess a logical structure:



(2)

This problem took a long time to solve because the structure of logic is not easily expressible as a mathematical structure. The perspective of **model theory** is helpful in understanding this connection:



(3)

I will first explain the structure of the neural side, then the structure of the logic side.

1 The structure of neural networks

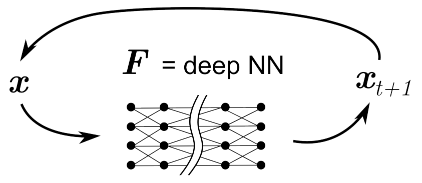
A **neural network** is basically:

$$\begin{array}{c} \textbf{weight}\text{ matrix for each layer} \qquad \text{total \# of layers} \\ F(\boldsymbol{x}) = \sigma(W_1 \sigma(W_2 \ldots \sigma(W_L\, \boldsymbol{x}))) \end{array} \tag{4}$$

where $\oint$ is the sigmoid function applied component-wise to each neuron (whose role is to provide **non-linearity**).

$\oint$ acts on each component of $\boldsymbol{x}$; Its action is **not invariant** under a change of coordinates. Therefore, $\oint$ is not a vector operation, and thus $\underline{\mathbb{X}\text{ is not a }\textbf{vector space}\text{ structure}}$. Common practice is to denote $\vec{x}$ as a vector, but this is misleading.

Consider, for example, to recognize *"white cat chases black cat"*. The *"cat"* object has to be recognied twice; Obviously we should not waste 2 neural networks to do this. If we connect a neural network **head to tail**, we get a minimalist cognitive architecture with a **recurrent** loop:



(5)

Its state equation is:

$$\boxed{\text{discrete time}} \qquad \boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t) \tag{6}$$

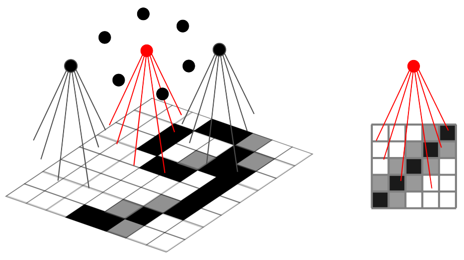
$$\boxed{\text{continuous time}} \qquad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{7}$$

From this we can see that $\mathbb{X}$ is a **differentiable manifold**. The deeper theory is that it is a Hamiltonian system, having a **symplectic** structure (cf. our first paper [12]).

1.1 What are "features"?

Now let's think about how a neural network performs pattern recognition, perhaps it would be illuminating....

Consider the simplest case, eg. a layer of neural network extracting visual features from the digit "9". This layer may have many neurons (left figure), each neuron locally covers a region of the input layer, the so-called "local receptive field" in the neuroscience of vision (right figure).



(8)

Suppose the red neuron is responsible for recognizing the "diagonal line" feature. Its equation is $\boldsymbol{y} = \oint(W\boldsymbol{x})$. The matrix W's role is to affine "rotate" the feature space, so that the features we desire is pointing in a certain direction. Then we use $\oint$ to "**squeeze**" the desired and undesired features. The output after $\oint$ represents the **presence or absence** of a particular feature, ie $\{0, 1\}$. This is a form of information **compression**.

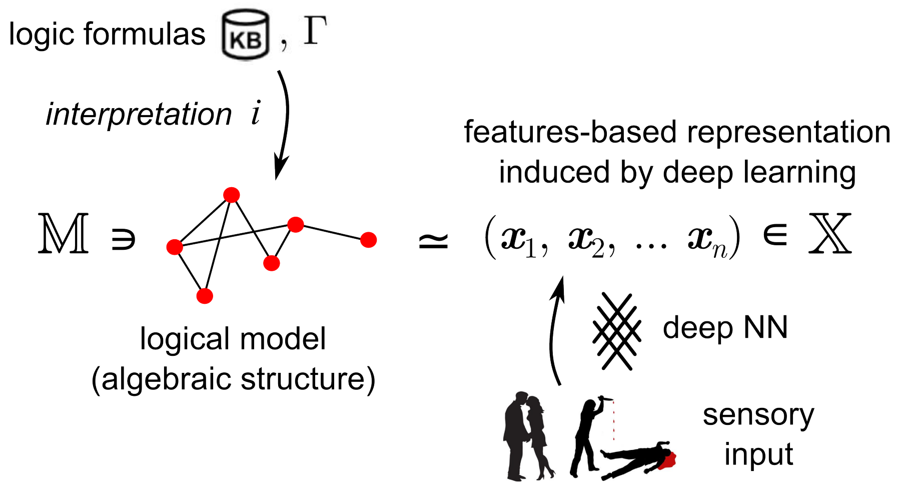
From the above reasoning, we may draw a general principle, which I call the **Neural Postulate**:

- Each neuron represents the presence or absence of a certain **feature**
- Higher-layer neurons represents **relations** between lower-layer features (9)

Following this line of thinking, we may conjecture this correspondence:

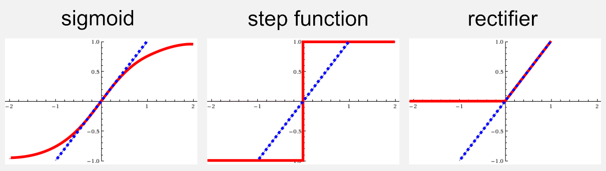
$$\begin{array}{rccl} & \mathcal{M} & \simeq & \mathcal{X} \\ \text{constant} & ● & \Leftrightarrow & \text{neuron} \\ \text{relation} & ●\!-\!● & \Leftrightarrow & \text{relation between higher and lower neurons} \end{array} \tag{10}$$

The following cartoon illustrates this correspondence from the perspective of **model theory**:


logic formulas
KB
, Γ
interpretation i
features-based representation induced by deep learning
𝕄 ∋
≃
(x₁, x₂, … xₙ) ∈ 𝕏
logical model (algebraic structure)
deep NN
sensory input

(11)

A little digression on chaos theory:
The role of $\textcircled{\int}^{-1}$ is to "stretch", dragging points that are close neighbors to more distant positions. Looking at the common **threshold functions**, we see they are all non-linear **deformations** away from the identity $y = x$:



(12)

This is very similar to the "horseshoe" proposed by Steven Smale [9], a recipe for creating chaos. A variant of the Smale horseshoe is the "baker map", analogous to kneading dough in bakery. "Stretching" and then putting back to the original space, and repeating this process, creates chaos [2] [10].

2 Structure of logic

A **logic system** can be defined as:

- a set of symbols for **constants**, **predicates**, and **functions**
- **propositions** built from the above atoms
- **connectives** between simple propositions, eg: $\neg, \wedge, \vee$
- The **logical consequence** relation: $\Gamma \vdash \Delta$

The learning algorithm for logic-based AI is studied under the topic of ILP (inductive logic programming), which is a well-established field (cf. my tutorial [11] or de Raedt's textbook [7]). ILP performs **combinatorial search** in the symbolic space of logic formulas, and is very slow. The gradient descent of neural networks is much faster. In the following we examine the prospects of combining these 2 approaches.

Our idea is to transfer the structure of logic to neural networks. The logic structure can be broken down into:

- propositional-level structure
  (the mental state $\boldsymbol{x}$ is represented as a **set** of propositions $p_i \in \boldsymbol{x}$)
- sub-propositional structure

2.1 Propositional-level structure

2.1.1 Decomposition of the mental state into propositions

The mental state $\boldsymbol{x}$ may be broken down into a conjunction of propositions:

$$\boldsymbol{x}_1 = \text{ I'm attending a seminar } \wedge \text{ I feel hungry } \wedge \ldots. \tag{13}$$

There could be another mental state:

$$\boldsymbol{x}_2 = \text{ I'm riding the subway } \wedge \text{ I feel hungry } \wedge \ldots. \tag{14}$$

It would be very inefficient if x_1 and x_2 are considered completely different states (without decomposition).

2.1.2 Boolean algebra

A **Boolean algebra** is a structure with:

$$\mathcal{B} = (\overset{\text{underlying set}}{A}, \overbrace{\wedge, \vee}^{\text{binary ops}}, \overset{\text{unary op}}{\neg}) \tag{15}$$

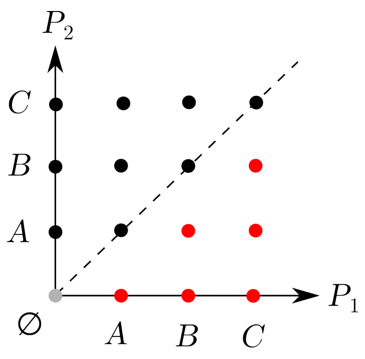
From my experience in logic-based AI, the most essential aspect of the Boolean structure (especially after introducing fuzzy-probabilistic reasoning) is this **commutativity** relation:

$$\forall a, b \in \mathcal{B}. \qquad a \wedge b = b \wedge a \tag{16}$$

2.1.3 Mapping propositions → vector space

Our gaol: map a set of propositions (*order is unimportant*) to a vector space.

Consider the simplest situation: P_1 and P_2 are two **containers** for propositions, the mental state is (P_1, P_2). Each proposition can be either A or B. We can arrange the placement of elements of $P_1 \times P_2$ in this way:



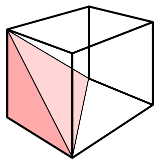
(17)

For example, the 3 red dots in the lower triangle ⊿ represent $\{A\}$, $\{B\}$, and $\{A, B\}$. The diagonal elements are not needed, because (A, A) etc. are *redundant* repetitions. In other words, all the propositional combinations are:

$$⊿ \setminus \text{ diagonal } \cup \varnothing \tag{18}$$

More abstractly, elements in this "symmetric space", eg. $(p_1, p_2, ..., p_n)$, are *invariant* under actions of the **symmetric group** S_n, n is finite.

In high dimensions, this space-saving scheme is extremely efficient: After symmetrization, a "corner" of the hypercube has a volume that is only $1/n!$ of the original cube, eg. $n = 3$:



(19)

Even as the **curse of dimensionality** grows exponentially, $n!$ is well sufficient to offset it.

Now consider the **functions** that live on this symmetric space; How are their structures affected? Before symmetrization, the functions have domains:

$$\boldsymbol{F} : \mathbb{X} \rightarrow \mathbb{X} \quad = \quad \mathbb{P}^n \rightarrow \mathbb{P}^n \tag{20}$$

where $\mathbb{P}$ is the space of a single proposition. After symmetrization:

$$\boldsymbol{F} : \text{sym}(\mathbb{P}^n) \rightarrow \text{sym}(\mathbb{P}^n) \quad = \quad ⊿ \rightarrow ⊿ \tag{21}$$

The symmetric space $\text{sym}(\mathbb{P}_1, \mathbb{P}_2, ...\mathbb{P}_n)$ possesses an **order**. That is to say, if the input to a function is $(p_1, p_2, ..., p_n)$, we need to **sort** the p_i's according to a certain order on $\mathbb{P}$, before calling $\boldsymbol{F}$ to calculate.

Sorting in $\mathbb{P}$ is simple, because each $p \in \mathbb{P}$ already has a **location** in vector space (that is learned via deep learning). $p_1 > p_2$ can be defined by a **cone** in the vector space, and then the order in $\mathbb{P}^n$ can be given by **lexicographic ordering**.

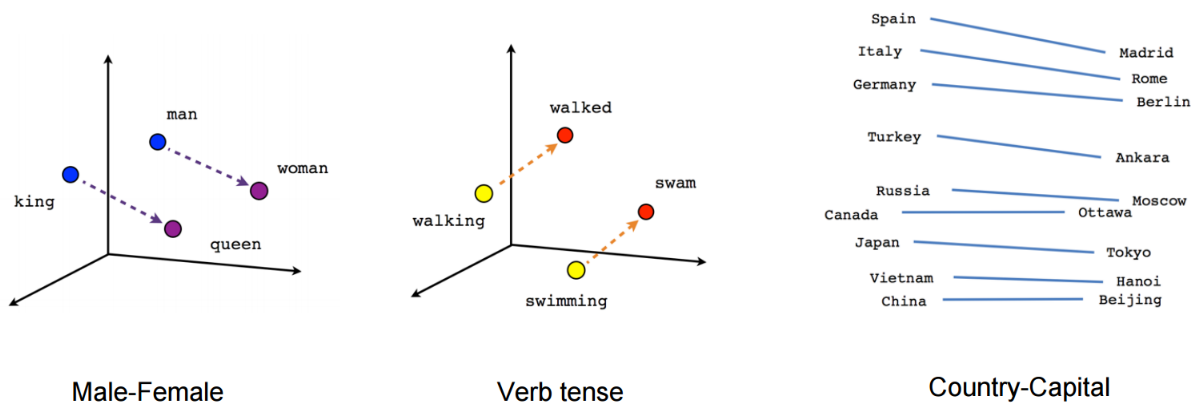
When implementing with as a neural network ⋇ : $\mathbb{R}^n \to \mathbb{R}^n$, we can keep the entire network, using only ⊿ as its domain and co-domain. It seems that we have gained nothing, but consider if we train the network with 100 samples on ⊿ versus the whole hypercube. ⊿ will receive the "full strength" of the data set whereas the hypercube will have its "firepower" diluted by $n!$.

2.2 Sub-propositional structure

This part is more complicated. Traditionally the structure of predicate logic is so esoteric from the point of view of mathematics that it hindered the progress of logic and logic-based AI.

2.2.1 Semantic distance and failure of compositionality

Word2Vec[6] embeds words into vector space with "semantic distance":



(22)

After Word2Vec, a natural next step is to represent **sentences** with semantic distance. One popular approach (eg [1]) is via **tensor products**. For example:

$$\text{\textit{“I love you”}} \quad \Rightarrow \quad \mathrm{i} \otimes \mathrm{love} \otimes \mathrm{you} \tag{23}$$

but this approach has serious problems: For example, pronouns such as "you" refer to various entities and must be resolved using rather complicated procedures, known as "abductive interpretation" in logic-based AI. Without such interpretations the word-entity correspondence becomes very inaccurate.

2.2.2 A logic based on "features"

Assuming that the compositionality problem can be bypassed, perhaps with logical atoms referring directly to entities (no pronouns, metaphors, etc), we may represent sentences by a "conjunction" of features (different from the $\wedge$ for propositions), eg:

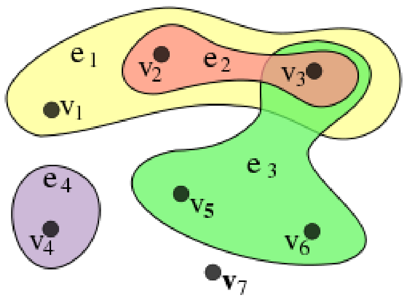
$$\begin{aligned} \boldsymbol{x} &= \textit{I feel hungry} \\ &= \textit{me} \cap \textit{physiology} \cap \textit{food} \cap \textit{negative-emotion} \cap \ldots \end{aligned} \tag{24}$$

This is like using a series of binary choices (= features) to determine an idea, like the game "20 Questions".

2.2.3 OpenCog's logic based on hypergraphs

To represent a complex scenario with many inter-relations, without using pronouns, perhaps **hypergraph** is a good choice. This is the idea behind OpenCog's representation.

A hypergraph is also called a "set system". Any hypergraph can be represented as a set of edges, each edge being an element of the power set of the nodes. Eg:



(25)

This representation is convenient for our purpose: Each edge can be represented as 1 proposition. The mental state $\boldsymbol{x}$ = a hypergraph = a set of propositions. We can use the symmetric space proposed above to represent the mental state.

Each proposition (edge) is a product of atomic concepts (nodes), we may further fix #(nodes) per edge to, say, 3:

$$\boxed{\text{proposition}} \qquad \boldsymbol{p} = c_1\ c_2\ c_3 \tag{26}$$

For example $c_1\ c_2\ c_3$ could be like subject-verb-object in natural language, where c_1 and c_3 are from the set of nouns / entities; c_2 from the set of verbs / relations. We don't need to be very precise about this, as the actual representation can be figured out via machine learning. This is the age of *post-modern*[⋆] AI!

2.2.4 "Linkage" phenomena in predicate logic

Next we consider **variable substitution**, a key feature of predicate logics. For example:

$$\forall X\ \forall Y\ \forall Z. \quad \text{grandfather}(X, Z) \leftarrow \text{father}(X, Y) \wedge \text{father}(Y, Z) \tag{27}$$

The links symbolize that the same variables must be substituted by the same objects, which is really the *essence* of **substitution**.

To duplicate this effect in neural networks, we can force a coordinate of the input space to be identified with another coordinate of the output space:

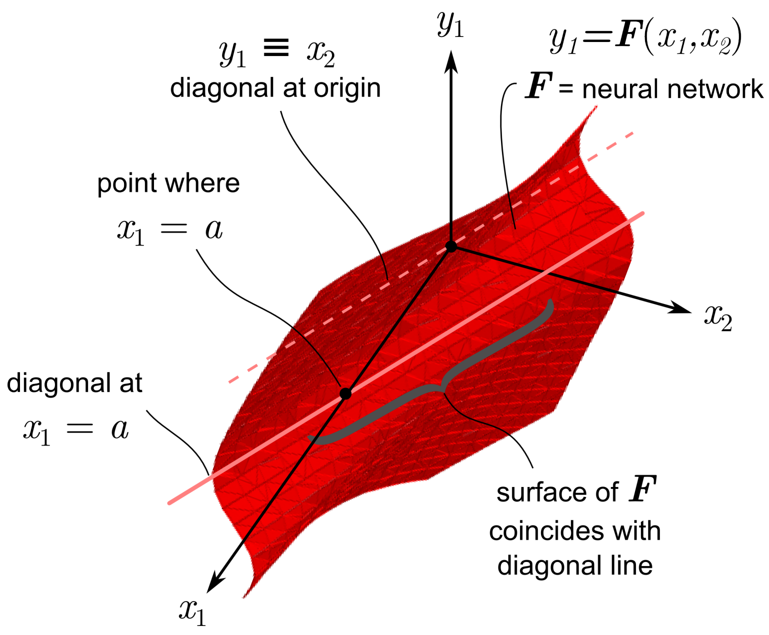
⋆ With "modern" meaning "structuralist", as in designating structural elements such as "NP (noun phrase)" in phrase-structure grammar or the "is-a" link in old-fashioned AI.

$$\begin{tikzcd}[column sep=small] \boldsymbol{F}:\ (x_1, \dots, x_i \arrow[rr, bend left, "id"] & , \dots, x_n) \mapsto (y_1, \dots, & y_j, \dots, y_n) \end{tikzcd} \tag{28}$$

But such identities do not always hold. They only exist at specific values of $\boldsymbol{x}$:

$$\boldsymbol{F}: \begin{cases} y_j \equiv x_i, & \text{if } \boldsymbol{x} = \hat{\boldsymbol{a}} \text{ for some coordinates except } x_i \\ y_h \equiv x_k, & \text{if } \boldsymbol{x} = \hat{\boldsymbol{b}} \text{ for some coordinates except } x_k \\ \text{... etc ...} \\ \text{is free otherwise} \end{cases} \tag{29}$$

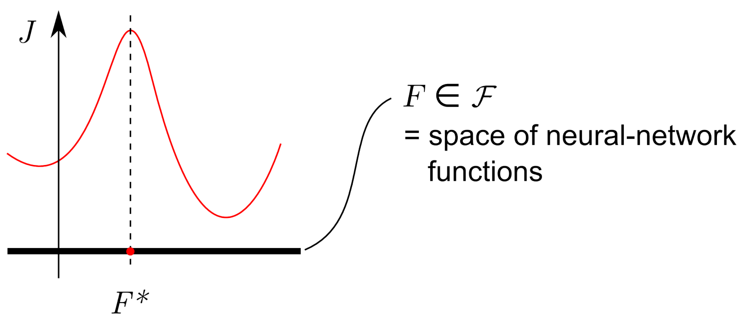
Geometrically, a linkage is a diagonal line that *restricts* the "graph" of the neural network function:



(30)

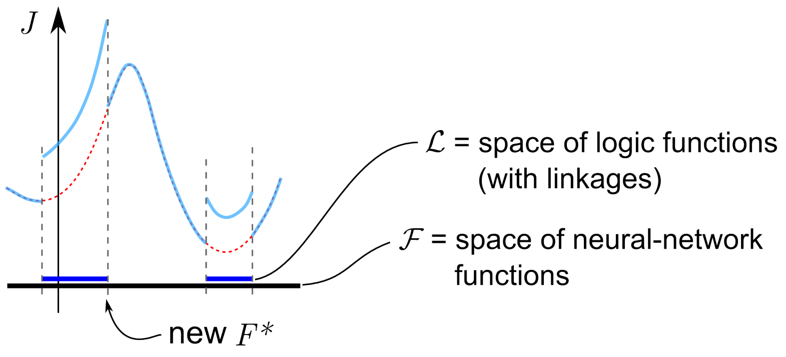
2.2.5 Learning algorithm with linkages

As an optimization problem, the goal of learning is to find the optimal neural network F^* in a function space (figuratively):



(31)

We need a new algorithm that gives priority (higher scores) to linkages:

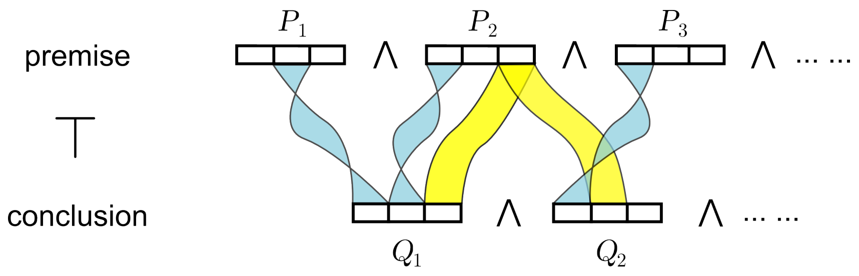


J
$\mathcal{L}$ = space of logic functions (with linkages)
$\mathcal{F}$ = space of neural-network functions
new F^*

(32)

Below is a figure that shows the "internal threads" of a function ($\vdash$) that maps a premise to a conclusion, ie:

$$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \quad \vdash \quad Q_1 \wedge Q_2 \wedge \ldots \tag{33}$$



(34)

where yellow "ribbons" represent id mappings (that respect coordinate positions), blue ribbons represent abitrary mappings.

As for learning algorithm, one idea is to *force* the learning of diagonals, whenever a potential identity function is detected. This creates a "propensity" for generalization. After a diagonal is learned, it may be destroyed by subsequent learning; which is fine. Also, the forcing of a diagonal may accidentally destroy previously learned knowledge. This may be balanced by **dual-lobe** re-normalization.

```
for each data item do
    detect if a potential identity may exist;
    if identity seems to exist then
        force-learn the diagonal;
    else
        learn as usual;
    end
end
dual-lobe re-normalization;
```
(35)

Appendix: logic and logic-based AI

This part is to be separated to become a tutorial paper...

A.1 Algebraic logic

Traditionally, there are 2 ways to express the **sub-propositional** structure of logic: via **predicate logic** and **relation algebra**. No matter which way, their essence is **variable substitution**. Interestingly, substitution is also the essence of "algebra", but here substitutions occur **implicitly**, so it is actually difficult to use algebra to express them explicitly.

As for **λ-calculus**, it is essentially also a scheme for managing substitutions. Whereas **combinatory logic**, equivalent to λ-calculus, eliminates the use of "variables". The price to pay is an increase in length of the formulas (more complex than relation algebra).

The algebraization of (first-order) predicate logic is given by Alfred Tarski's **cylindric algebra**:

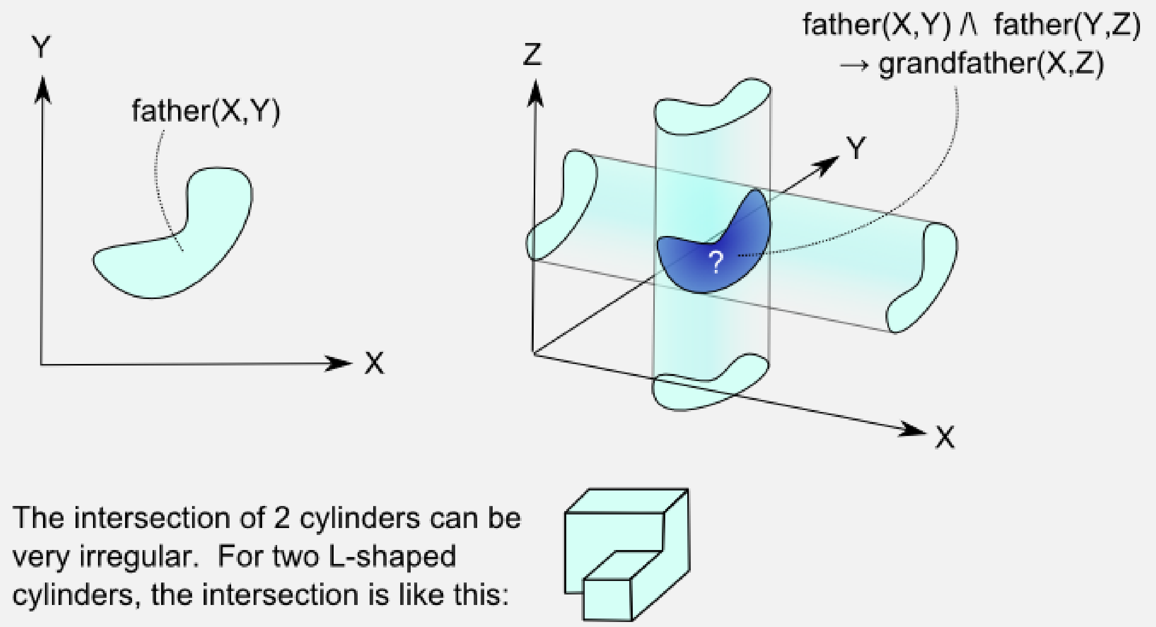
$$\frac{\text{propositional calculus}}{\text{Boolean algebra}} = \frac{\text{predicate calculus}}{\text{cylindric algebra}} \tag{36}$$

The algebraization of **higher-order logic** (HOL) is provided by **topoi theory** [3] [4]:

$$\frac{\text{intuitionistic propositional calculus}}{\text{Heyting algebra}} = \frac{\text{higher-order logic}}{\text{elementary topos}} \tag{37}$$

HOL is equivalent to **untyped** λ-calculus, whereas the **typed** λ-calculus (= type theory) is somewhat weaker. The characteristic of the entire HOL family is "substitution of equal by equals", which causes the length of formulas to increase. But predicate logic performs substitution with **objects**, so it does not increase the length of formulas; However, its algebraization introduces notions such as cylindrification and the diagonal set. For example, in fig. (30) there is the diagonal $y_1 \equiv x_2$.

This is an illustration of **cylindrification**:



The intersection of 2 cylinders can be very irregular. For two L-shaped cylinders, the intersection is like this:

(38)

A.2 Relation algebra

以前曾经对 relation algebra [8] [5] 有些憧憬，因为它比较接近人类自然语言，但其实 relation algebra (RA) 和 first-order logic (FOL) 基本上是等效的，在 FOL 里面有 linkage 的复杂性，但在 RA 里面这个复杂性其实也没有消失。可以说「复杂度是守恒的」[a]。

举例来说，在 (27) 中表达「爸爸的爸爸是爷爷」，可以用 RA 更简单地表达：

$$\text{father} \circ \text{father} = \text{grandfather} \tag{39}$$

实际的推导是这样的：

$$\begin{aligned} \text{John father Pete} \\ \text{Pete father Paul} \\ \text{John father} \circ \text{father Paul} \end{aligned} \tag{40}$$

这时要 代入 上面的等式才能得出结论：

$$\text{John grandfather Paul} \tag{41}$$

所以 linkage 的复杂性变成了代数 formula 长度的复杂性。

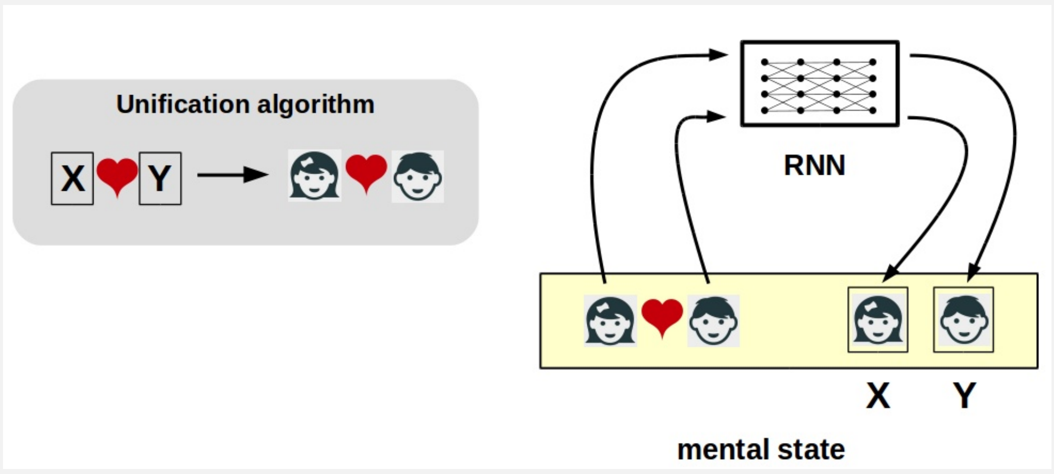
A.3 Fractal structure

「长度」本身没有不妥，但我们的目的是将逻辑式子嵌入到状态空间 $\boldsymbol{x}$ 里，这时 variable length 令人很头痛，但 fixed length 没有此问题。如果硬要将 variable length 的式子嵌入到（有限维）向量空间中，似乎必须用到fractal 结构（因为它有自相似性），同时需要设计一种新的神经网络，它先天性地有 fractal 结构在里面；但这比较复杂，我没有在这方向 explore。

A.4 Trading space with time

上面说的是 substitution 在空间中的 linkage 结构。但也可以将 substitution **分拆**成若干个简单的步骤。方法是：将某些内容放进记忆体中的「盒子」中，这些盒子充当 variables 的角色，也就是很具体地实现 substitution 的动作。换句话说，将**空间复杂**

性转换成**时间复习性**[b]；以下是示意图：



(42)

X，Y 是状态空间 $\mathbb{X} \ni \boldsymbol{x}$ 里面的「盒子」= variables。

先前我说的 linkage 结构，放在有限维向量空间比较方便，但它是 first-order logic，处理 higher-order 关系时有很大困难。Higher-order relations 似乎还是要用这种分拆步骤的方法解决。

A.5 Cartesian-closedness

折腾了这么久，但仍然缺少了一个重要的特性：Cartesian-closedness。它指的是在某个範畴内，对任意的 A, B，都必然可以找到它们的：

$$\boxed{\text{product}}\ A \times B \quad 和 \quad B^A\ \boxed{\text{exponentiation}} \tag{43}$$

In logic this refers to:

$$A \wedge B \quad 和 \quad A \to B \tag{44}$$

为什么需要 Cartesian-closed？注意在 minimal architecture 里，只有两种记忆，即瞬时记忆 $\boldsymbol{x}$ 和永久记忆 $\boldsymbol{F} =$ KB $\vdash$。从 logic-based AI 的角度来看，KB 里装著 logic rules，但 $\boldsymbol{x}$ 里面只有 ground facts。Ground sentence 指的是**没有变量**的命题，例如：

$$\boldsymbol{x} = 见到地上有血迹 \tag{45}$$

相比之下，logic rule 是有变量的 conditional statement，例如：

$$\forall Z. \quad Z\ 有血迹 \to Z\ 可能是凶案现场 \tag{46}$$

如果在 $\boldsymbol{x}$ 里面可以存放 rule，那表示 $\boldsymbol{x} \ni \mathbb{X}$ 是一个 Cartesian-closed category。这样的 $\mathbb{X}$ 是一个更 powerful 的结构，亦即系统可以思考更复杂的 thoughts。更重要的是，$\boldsymbol{x} \ni \mathbb{X}$ 和 $F =$ KB $\vdash$ 现在**地位平等**，因为它们都是 $\mathbb{X} \to \mathbb{X}$ 的**函数**，因为 Cartesian-closed 表示 $\mathbb{X} \simeq \mathbb{X}^{\mathbb{X}}$。这个特性在 belief revision 中似乎会很有用（见下节）。

如何在神经网络中做到 Cartesian-closed？记得神经网络中 $\boldsymbol{x}$ 是一组 features $= (x_1, x_2, ..., x_n)$。一个办法是将所有 $\boldsymbol{x}$ 都变成 $\mathbb{X} \to \mathbb{X}$ 的函数。逻辑的 implication

$A \to B$ 显然是函数，但单一 ground sentence A 也可以是函数：$\top \to A$ ，其中 $\top$ 是逻辑「真」。注意：这些函数中可以有 linkages，即处理变量的能力。

转到神经网络中，$A \to B$ 是一个 $\mathbb{X} \to \mathbb{X}$ 的神经网络。$\top \to A$ 也是一个 $\mathbb{X} \to \mathbb{X}$ 的神经网络，但它将 $\mathbf{1} \mapsto \boldsymbol{A}$。

那 $\mathbb{X}$ 是一个怎样的空间？它本身可以是一个深度神经网路的 weights，但这个神经网络的输入 / 输出层必须有足够的**阔度**去处理**它自己**！表面上似乎不可能.... 越多的 weights 需要更多的 weights 去处理.... 但如果令 $\boldsymbol{x}$ 储存一些 partial functions 或许可以。

A.6 Application to belief revision

Belief revision （也可以叫 “truth maintenance”）是经典逻辑 AI 发展的高峰。如果可以用我们的新 architecture 做到 belief revision，我们会很有信心这个理论是 general intelligence。

正常的逻辑运作模式是由 $\boxed{\text{KB}}$ 作用在 $\boldsymbol{x}$ 上给出新的 $\boldsymbol{x}$：

$$\boxed{\text{KB}}(\boldsymbol{x}) = \boldsymbol{x}' \tag{47}$$

但 belief revision 或者可以看成是 $\boldsymbol{x}$ 作用在 $\boxed{\text{KB}}$ 上的结果：

$$\boldsymbol{x}(\boxed{\text{KB}}) = \boxed{\text{KB}}' \tag{48}$$

由於 Cartesian-closedness，$\boldsymbol{x}$ 和 $\boxed{\text{KB}}$ 的地位是平等的，使上面的运作成为可能。

这只是一个 vague idea，我会再回到这里填补这空白....

[a] 这句话是 category theorist Eugenia Cheng 说的。
[b] 人脑的脑电波由 0.5 Hz 到 40 Hz 都有，我们感觉上瞬间的动作可能已经过了若干次迴路。

Acknowledgement

Thanks to Ben Goertzel for discussions on the AGI mailing list. Ben first pointed out the advantage of neural network learning over inductive logic learning, which prompted me to research their relationships. Thanks also to Juan Carlos Pinto for personal discussions on the structure of neural networks.

References

1. Coecke, Sadrzadeh, and Clark. Mathematical foundations for distributed compostional model of meaning. *Linguistic Analysis 36, 345-384*, 2010.
2. Robert Gilmore and Marc Lefranc. *The topology of chaos: Alice in stretch and squeezeland.* Wiley-VCH, 2011.

3. Joachim Lambek. Categorical versus algebraic logic. In Andréka, Monk, and Németi, editors, *Algebraic logic*, pages 351–360. North-Holland, 1988.

4. Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic – a first introduction to topos theory.* Springer, 1992.

5. Roger Maddux. *Relation algebras.* Elsevier, 2006.

6. Mikolov, Sutskever, Chen, Corrado, and Dean. Efficient estimation of word representations in vector space. *Proceedings of workshop at ICLR*, 2013.

7. Luc De Raedt. *Logical and relational learning.* Springer, 2008.

8. Gunther Schmidt. *Relational mathematics.* Cambridge, 2010.

9. Stephen Smale. Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 1967.

10. Tamás Tél and Márton Gruiz. *Chaotic dynamics: an Introduction based on classical mechanics.* Cambridge, 2006.

11. King Yin Yan. ILP tutorial `https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/genifer/Genifer-induction(30July2012).pdf`.

12. King Yin Yan, Juan Carlos Kuri Pinto, and Ben Goertzel. Wandering in the labyrinth of thinking – a cognitive architecture combining reinforcement learning and deep learning. (to be submitted AGI-2017).